FIDES: An Advanced Chip Multiprocessor Platform for Secure Next Generation Mobile Terminals

INOUE, Hiroaki†, Akihisa Ikeno‡, Masaki Kondo‡, Junji Sakai† and Masato Edahiro†

† System Devices Research Laboratories, NEC Corporation
1120, Shimokuzawa, Sagamihara, Kanagawa, 229-1198 Japan

‡ NEC Informatec Systems, Ltd.
3-2-1, Sakado, Takatsu-ku, Kawasaki, Kanagawa, 213-0012 Japan

h-inoue@ce.jp.nec.com, a-ikeno@pb.jp.nec.com, ms-kondo@pb.jp.nec.com,
jsakai@bc.jp.nec.com, eda@bp.jp.nec.com

ABSTRACT

We propose a secure platform on a chip multiprocessor, known as FIDES, in order to enable next generation mobile terminals to execute downloaded native applications for Linux. Its most important feature is the higher security based on multi-grained separation mechanisms: coarse-grained processor-level separation of the basic-function domain from other domains for such downloaded applications, medium-grained OS-level separation, and fine-grained process-level separation within SELinux. Four new technologies, which include three enhancements to SELinux, support the FIDES platform: 1) bus filter logic for processor-level separation can be implemented as a small logic, 2) XIP kernels for memory-efficient OS-level separation can reduce memory requirements by 182%, 3) policy separation for enhanced process-level separation can apply policies 2.1 times faster at system boot-up, and 4) dynamic access control can provide secure Inter-Domain Communications (IDCs) with an overhead of only 4% for IDC system calls. We implemented SELinuxes on an ARM-based multiprocessor. Therefore, the best-suited platform to secure next generation mobile terminals is the FIDES platform, which can provide higher security as well as higher performance and lower power consumption on chip multiprocessors leading the current technology trend of microprocessors.

Categories and Subject Descriptors

C.1.4 [**Processor Architectures**]: Parallel Architectures – *mobile processors*; D.4.6 [**Operating System**]: Security and Protection – *access controls*;

General Terms

Design, Security

Keywords

Secure Mobile Terminal, Chip Multiprocessor, Linux

CODES+ISSS'05, Sept. 19–21, 2005, Jersey City, New Jersey, USA.

1. INTRODUCTION

Next generation mobile terminals are likely to require a dedicated execution environment in which users are able to use native applications downloaded from open networks. This means that application processors will have to offer higher performance and lower power dissipation. Current application processor architectures, however, are based on heterogeneous multi-cores, such as CPUs and DSPs [6]. Since a high clock frequency is needed to increase CPU performance, this makes it difficult to reduce power dissipation. One promising approach would be to use a chip multiprocessor like MP211 System-on-a-Chip (SoC) [1] as an application processor. With multiprocessors, the desired level of performance is achieved with a number of processors that operate at moderate clock frequencies, which helps to keep power consumption low. It should be noted that the area requirement of such multiprocessors would not be a problem with fabrication technology of 90 nm or better [14].

Next generation mobile terminals will also require OSs with higher functionality in order to reduce the cost of software development. Currently, real-time OSs, such as μ-ITRON [9], and specialized OSs, such as Symbian OS [13], are commonly used because they perform effectively on mobile terminals. However, the number of software engineers capable of working effectively with such OSs is limited due to the closed nature of the community dedicated to the development of mobile terminals. Linux OS, with its large and open community, would be a good candidate for addressing this problem [5].

Finally, the downloading of native applications will mean that security will become an increasingly important issue. That is, it will be important to be able to confine the influence of such downloaded applications to a certain level, and NTT DoCoMo, IBM, and Intel have, in fact, jointly announced specifications designed to encourage the development of mobile terminals having such security capability [11].

One approach is to verify applications before making them available for downloading to mobile terminals. This is the approach taken by BREW [4]. Since the verification is conducted on an actual mobile terminal, its results may be considered quite reliable.

Another approach is to confine applications which might contain vulnerabilities within a virtual domain (*i.e.,* into a so-called

sandbox). Such virtual machines as Java [8], VMware [15] and User Mode Linux (UML) [7] can be used to create a virtual domain for an application to be downloaded, so that the application's execution environment can be kept separate from the system itself. This approach appears to the one best suited to establishing future higher-level security, since future applications can be expected to become too complex for perfect pre-verification.

Here, we propose a secure platform which employs Linux on a chip multiprocessor in order to achieve higher performance and more secure execution environment on OSs with high functionality. We call this platform FIDES. It forms a domain on each processor, so that the basic-function domain is separated physically from the other domains. This platform can confine the influence of any downloaded application to the single domains on which it is to be executed. Further, since, unlike the second approach, applications are not confined within a virtual domain, the CPU performance can be kept high. In this way, FIDES with multi-grained separation mechanism is able to offer high performance and secure execution environment on high-functionality OSs; the features are needed for next generation mobile terminals.

The remainder of this paper is structured as follows: Section 2 gives details on the technologies used to build the FIDES platform; Section 3 shows presents an evaluation of platform performance; and Section 4 discusses conclusion and future work.

2. FIDES PLATFORM

The basic structure of the FIDES platform is outlined in Figure 1. Here, we assume the use of an asymmetric multiprocessor in which all caches are independent.

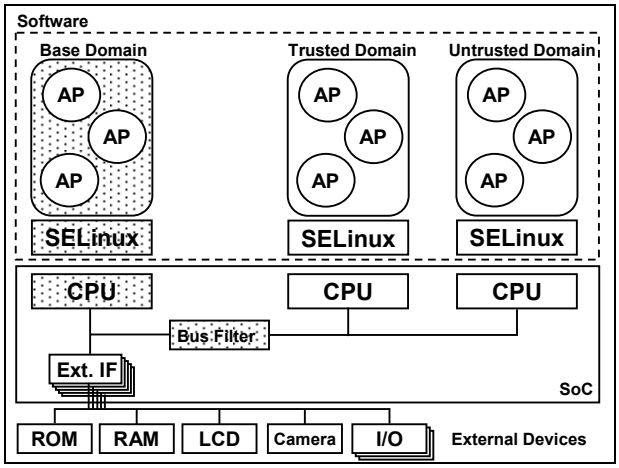

Figure 1: FIDES Platform with Multi-grained Separation Mechanisms for Secure Mobile Terminals

The basic FIDES platform has three domains, *i.e.*, base, trusted, and untrusted. The base domain contains the basic functions of a mobile terminal, such as a mailer and a browser. Downloaded applications are never executed on the base domain, which is only assumed to be sufficiently trustworthy to handle the basic functions of the mobile terminal. Downloaded applications validated by communication carriers as being trustworthy are executed on the trusted domain. All other downloaded applications are executed on the untrusted domain. In this way, the base domain is protected from the behavior of all applications on other domains. Therefore, this platform has better potential ability to realize a secure mobile platform with high performance and low power consumption. Note that the number of the domains on FIDES platform is not limited to only three and can be flexibly determined by system requirements.

However, in order to put this FIDES platform to practical use, the platform needs two important components: hardware support for coarse-grained processor-level separation and secure OSs for medium-grained OS-level separation and fine-grained process-level separation.

2.1 Processor-level Separation Logic

The FIDES platform embeds a processor-level separation logic into the system bus, such as AMBA [2]. This logic provides a new mechanism, which is based on the access matrix between bus masters and bus slaves at the bus level. An address range checking mechanism is also required, since some memory resources on different regions, such as SDRAMs or Flash ROMs, are accessed from bus masters. The abstract structure of bus filter logic is illustrated in Figure 2, where the bus signals have been defined in AMBA. The basic role of this logic determines whether an access from a bus master to a bus slave should be granted or not. The decision is based on the access matrix, which stores information in which all masters can have read and write access to an I/O or an address range of memory.

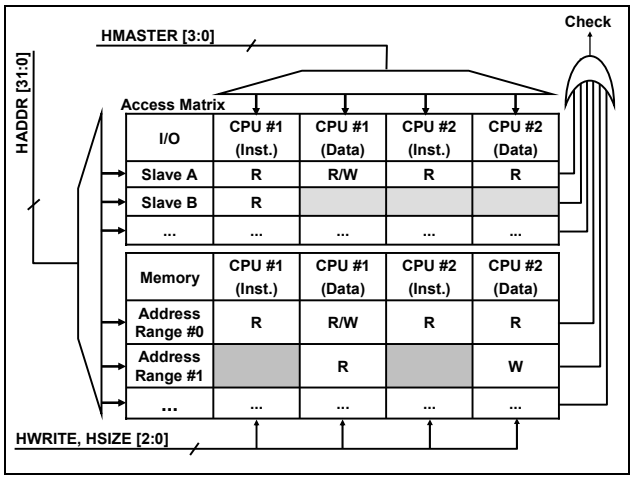

Figure 2: Bus Filter Logic – New Logic for Processor-level Separation on a Chip Multiprocessor

The content for the access matrix should only be set by the processor on the base domain. Also, the default setting is preferable to denying all access by other processors to base-domain resources, which must be protected.

The trade-off in terms of the design is the number of address-range entries, which determines whether an access to a memory can be permitted. The more this number increases, the longer access latency is. Therefore, the number of the entries should be close to the number of protected resources. Also, if the access latency and the logic area are sufficiently small for the system, MMU architecture might be preferable. This is because it can

provide a simple mechanism of virtualization to the resources as well as access control to the bus slaves.

2.2 Enhanced Secure OS

Three technologies enhance Security-Enhanced Linux (SELinux) [10] for fine-grained process-level separation: eXecute-In-Place (XIP) kernels to reduce total memory requirements; policy separation, which can be independently applied to individual domains; and dynamic access control, by which the access rights of processes can be changed dynamically to prevent the propagation, through Inter-Domain Communication (IDC), of viruses or other malicious attacks.

2.2.1 XIP Kernels for OS-level Separation

Processor-level Separation can provide OS-level separation for a system without any modification of an OS. However, the available resources in embedded systems are limited by system cost. An example of such resources is memories, which include ROMs and RAMs. Therefore, total memory requirements have to be reduced in order to support OS-level separation, which causes the system to run multiple OSs.

In general, read-only data is often placed on ROMs to reduce the total memory requirement for RAMs or to shorten the boot time for the system, since data does not need to be copied in RAMs. Typical read-only data are instructions, which should be executed directly on a ROM. This technology is usually called XIP [3]. Currently, the XIP technology is supported only by a single processor. Therefore, we developed a new XIP technology for the FIDES platform, called as XIP kernels, which is improved from a traditional XIP technology for a single processor.

The concept behind XIP kernels is illustrated in Figure 3. Read-only sections, such as instruction and read-only data, are collected together as one continuous section. This is achieved by modifying the linker script. In this way, read-only sections are retained in the ROM. The other sections are copied to the RAM. On the other hand, SELinux's data section is copied to the dedicated working area on the RAM, which is determined by the processor ID on the SELinux. Each virtual address space in the XIP kernels seems to be the same as a single kernel.

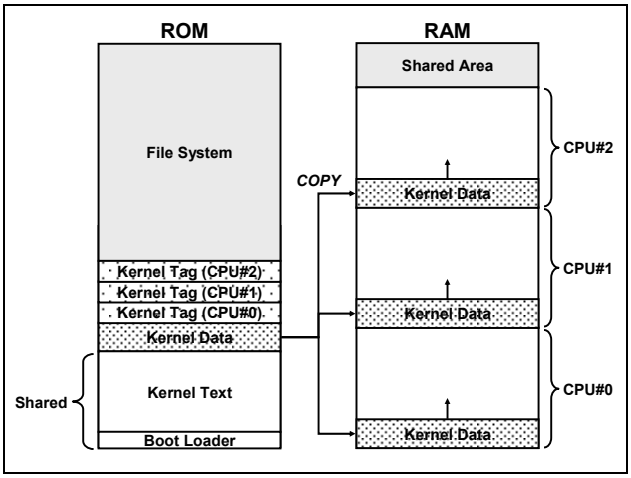

Figure 3: XIP kernels for Memory-efficient OS-level Separation

2.2.2 Policy Separation for Process-level Separation

SELinux's policy is a key of system security to realize the process-level separation. Only a policy can control the access rights of processes. Note that the access right of a process specifies which system calls the process can execute. However, the larger the policy becomes, the longer the execution time for system calls is. This degraded performance would not be acceptable for embedded systems. The policy should be as simple as possible to reduce such overhead. Therefore, the FIDES platform separates a single policy into three to simplify the policy for each domain. For example, the policy for the base domain may be relaxed, the policy for the trusted domain may have moderate constraints, and the policy for the untrusted domain may have strict constraints, since the system calls for untrusted applications, which may have a bad influence on the other domains, should be restricted.

We developed a new mechanism for policy separation for SELinux kernels (see Figure 4). This mechanism can be achieved by modifying the SELinux code to load the policy file corresponding to the processor ID. Further, it provides two additional effects for the system. First, the time to boot up SELinux can be reduced, since the kernel loads a smaller policy file. Second, simpler policies for each domain can reduce the overhead to execute system calls.

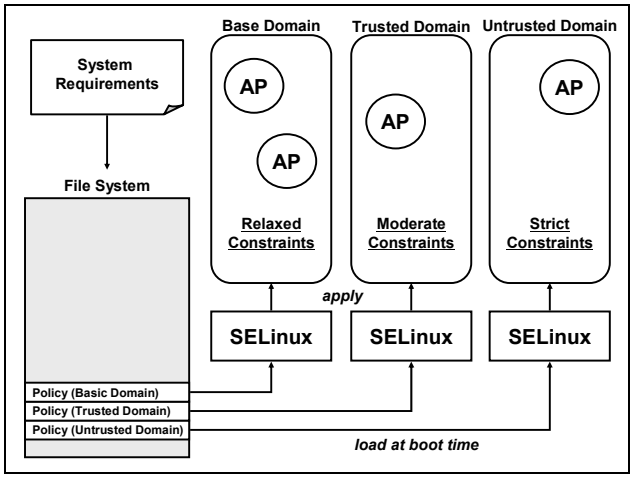

Figure 4: Policy Separation for Enhanced Process-level Separation

2.2.3 Dynamic Access Control for Secure IDCs

IDC plays an important role to provide useful services for embedded systems. For example, an application on the trusted domain could download a music file through IDC with basic-function applications on the base domain. However, such an IDC mechanism might create security holes. A process on a trusted domain, which communicates with another process on an untrusted domain, might be attacked through malicious data propagating from the process on the untrusted domain. Therefore, the concept of secure IDCs is important for the systems.

The simplest approach would be to verify all applications with IDC. However, this is not realistic unless such applications are sufficiently small to be verified formally. If applications with IDC cannot be pre-verified, they should be controlled dynamically. One approach is with a tainted mode of Perl, which prohibits

system calls that use data acquired from communication with an untrusted domain as an argument. It can be efficiently implemented for interpreter languages, such as Perl, since malicious data can easily be traced in interpreters. However, it is impossible to apply to applications written in native languages like C, since C specifications make it difficult to trace data. Note that such native languages are our target in order to support the downloading of native applications, as mentioned in Section 1.

Therefore, we developed dynamic access control for SELinux to provide more secure IDCs, which is illustrated in Figure 5. This mechanism can dynamically change the access rights of processes related to IDCs into the limited access rights. For example, two processes are doing IDC from domain B to domain A in the figure. The access right of the application on domain A is then changed to a stricter one. On the other hand, the application on domain B may have the original access right, since the application has not received malicious data. Note that the access right of a process specifies which system calls the process can execute.

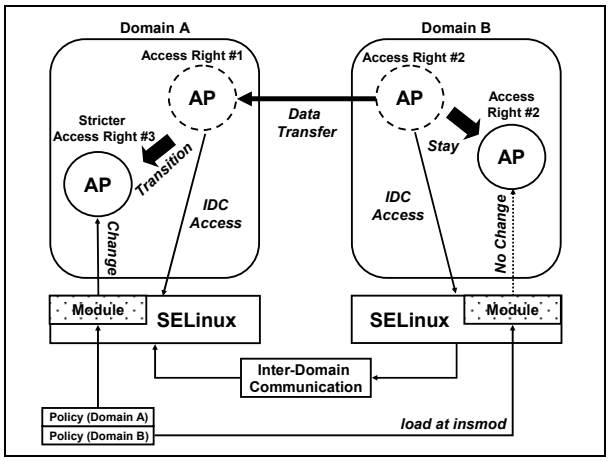

Figure 5: Dynamic Access Control – New Mechanism for Secure IDCs

Currently, sockets and files, which are commonly used for inter-processor communication, are available as IDCs of the FIDES platform. Then, this mechanism can be implemented as a kernel module, which hooks only two security check functions of SELinux: inode_permission to check open system call for files and socket_bind to check bind system call for sockets in the structure of security_operations. When this module is loaded into the kernel, it reads two new policy files. These files have information on changed access rights, and information on managed kernel objects, which include file names or socket information with the address family (*e.g.* AF_INET), type (*e.g.* SOCK_STREAM), protocol, and port number.

Note that there is only one concern with this approach. It is extremely difficult to determine when the original access right of a process must be restored, after the access right has been once changed. The process might execute system calls that use malicious data stored in its own process space. In terms of this concern, the downloaded application which has already communicated with other domains should be killed or shut down at appropriate time. The mechanisms for such a downloaded application can be achieved, since mobile terminals has already had such handling mechanisms for non-native applications.

3. EVALUATION

3.1 Evaluation Environment

The evaluation environment is summarized in Table 1. We use a mobile application processor, known as MP211 SoC [1] of NEC Electronics Corporation. The main feature of this chip is a chip multiprocessor architecture with three ARM processors. We implemented SELinuxes on an ARM-based multiprocessor

Table 1: Evaluation Environment

Item	Feature
SoC	MP211
CPU	ARM926EJ-S x 3
Cache	I: 16KB, D: 16KB
Frequency	ARM: 200MHz, Bus: 100MHz
Memory	ROM:64MB, RAM:64MB
Linux	Embedded Linux 2.4.20 with SELinux

3.2 Qualitative Comparison

Table 2 lists the qualitative advantages offered by the FIDES platform over other platforms: software-only platform and new-mode platform. The most highly desired characteristics are underlined in the table. The information of secure IDCs is not shown in the table, since the mechanism of the other platforms is unclear.

In general, the effective way to evaluate the quantitative security level of a system design is unestablished, since it depends on the system implementation. Therefore, the magnitude of interaction to a protected domain, such as base domain, could be one candidate for the approximate metrics. This means that a platform with multi-grained separation mechanisms, which can reduce the interaction to a protected domain, would be more secure.

Table 2: Qualitative Comparison with Other Platforms

Feature	Software Only	New Mode	FIDES
Separation	Process	Process, OS	Process, OS, Processor (Section 2.1, 2.2.1-2)
Security	Low	Medium	High
Secure IDC	---	---	Dynamic Access Control (Section 2.2.3)
Performance	Low	Moderate	High (Section 2.2.2)
Power	High	Moderate	Low
Flexibility	High	Moderate	Moderate
HW Area	Small	Medium	Large
Reboot Time for Crashed Domains	High	High	Low (Section 2.2.2)

The software-only platform, whose security is based on only process-level separation on a secure OS, has the smallest hardware area and the highest flexibility of the three platforms. However, the security level is not so high, since a process might be able to exploit a vulnerability of the OS and the process could attack the base domain through the exploited OS.

The new-mode platform, whose security is based on process-level and OS-level separation, features a CPU capable of switching

operational modes [1] between "normal", and "secure." Basic functions are operated exclusively in the secure-mode, whereas operations for downloaded applications are conducted exclusively in the normal mode. In this way, basic-function operations are protected from any possible malicious-application content. This OS-level separation is supported by additional hardware logic. However, it does suffer from certain disadvantages for embedded systems. Notably, the platform needs the excessive length of time in order to reboot a crashed domain without any performance influence to the base domain. Further, the security level is not high enough, since a process might be able to waste a lot of CPU time by the execution code of busy-wait loop and the process could suppress the work of basic-function applications on the base domain indirectly.

Finally, FIDES platform with multi-grained separation mechanisms, which include the processor-level separation, offers the best security. In other words, this platform is more resilient to the attacks which consume the system resources, such as CPU time or memory. Further, the platform can reboot a crashed domain quickly without any performance influence to the base domain, since a processor for the crashed domain is independent on the processor for the base domain. This feature is more preferable for embedded systems. FIDES has the largest hardware area of the three, as previously discussed, whereas this would not be a problem with a fabrication technology of 90nm or better.

From the above comparison, the proposed FIDES platform is more superior as a secure platform for embedded systems, such as mobile terminals. In the remainder of this section, new four technologies to support FIDES platform are evaluated.

3.3 Bus Filter Logic

The estimated number of transistors for bus filter logic as a processor-level separation is less than 20 K. According to ITRS2003 [14], the transistor density logic (M transitors/cm^2) of an MPU is 77 in the hp90-nm fabrication technology node that appeared in 2004 or later. This is sufficiently small, since the area bus filter logic requires is less than 0.026 mm^2. Further, careful circuit implementation can reduce its latency overhead and power consumption.

3.4 XIP kernels

The effect of reducing the total memory requirement which XIP kernels gives to the system is shown in Figure 6, where the amount of memory for a single kernel without SELinux is normalized to 1, and the kernel configuration has been set to the minimum functions for the MP211.

The total memory requirements which include RAM and ROM are increased by 7%, when SELinux is introduced to the base kernel for a single processor. SELinux can be adapted to the system, since the memory overhead is very small. However, non-XIP kernels for chip multiprocessors make the memory requirements increase by 214%. On the other hand, XIP kernels can reduce the memory requirements by 182%, compared with non-XIP kernels. It is shown that this technology is very advantageous, since the shared text section on ROM is dominant in all sections of a Linux kernel. Also, this memory overhead is acceptable enough from the point of view of current memory trends, since the memory overhead for XIP kernels is suppressed to only 32%.

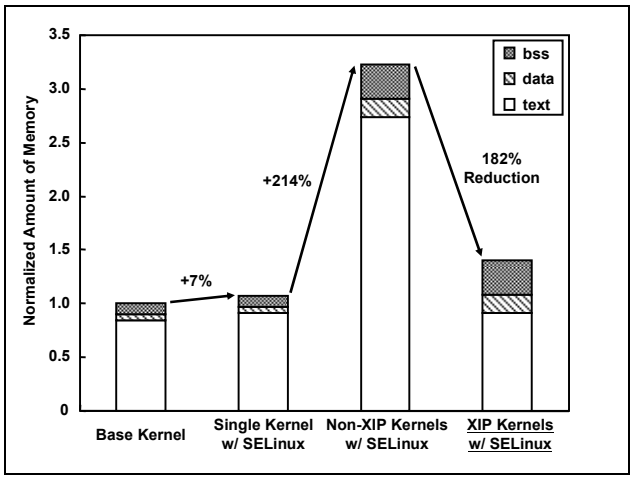

Figure 6: Memory Requirements Reduced by XIP kernels

3.5 Policy Separation

The main effect of policy separation is to simplify policy rules. It is difficult to evaluate these effects numerically. Instead, we evaluate the load time of a policy file. It has a great effect on reducing the boot time. This can be seen in Figure 7, where the number of rules in a policy file is along the horizontal axis, and the load time, which is an average of 20 measurements, for a policy with 352 rules is normalized to 1 along the longitudinal axis.

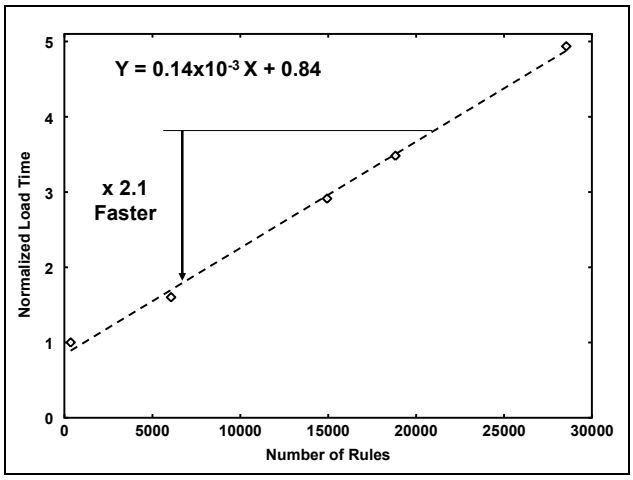

Figure 7: Faster Load Time for Policy File

Least-squares fitting reveals that the relation between the number of rules and load time is linear. The number of rules depends on the system requirements. In an example for our evaluation environment, the policy file for a single domain has 21,000 rules. Then, each policy file for three domains has less than 7,000 rules. Consequently, the load time for three domains would be 2.1 times faster. It should be noted that this load time cannot be achieved with a single processor under the same power consumption conditions, since the frequency is limited to a 1.5 times increment at maximum from our approximate calculation.

3.6 Dynamic Access Control

We evaluate the module size and the overhead to the execution time of system calls hooked by dynamic access control. The total size, which contains text section, data section and BSS section, is only 3.7 KB. This size is small enough for embedded systems.

In addition, the overhead to the execution time of hooked system calls, such as open system call and bind system call, is plotted in Figure 8, where the execution time is an average of 100 system calls, and the execution time on SELinux without this module is normalized to 1. The overheads of both system calls are only a few percent, when the access right of a process has not been changed. Also, the overheads are only 4%, when the access right has been changed.

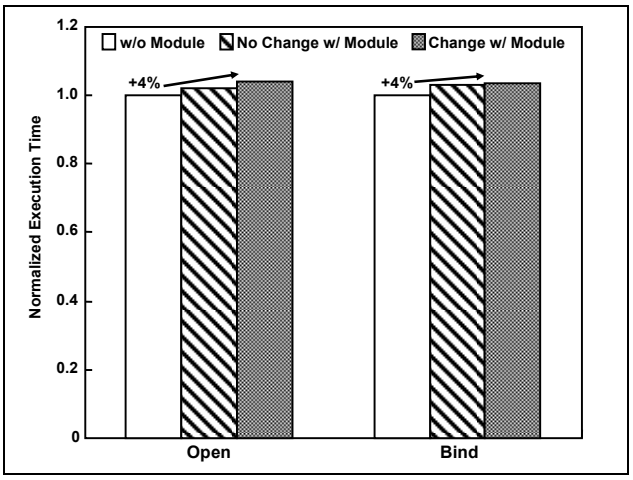

Figure 8: Small Overhead to Execution Time of System Calls Hooked by Dynamic Access Control

4. CONCLUSION

FIDES can provide an excellent platform for secure next generation mobile terminals based on the flow of mobile-terminal security. The FIDES platform, which employs Linux on a chip multiprocessor, could guarantee security in executing downloaded native applications.

The main feature is the higher security based on multi-grained separation mechanisms: coarse-grained processor-level separation of the basic-function domain from the other domains, medium-grained OS-level separation, and fine-grained process-level separation on SELinux as a secure OS. Bus filter logic for processor-level separation can be implemented as a small logic. Further, we enhanced SELinux in order to support the other-level separations and meet the embedded systems requirements. XIP kernels for memory-efficient OS-level separation can reduce memory requirements by 182%, policy separation for enhanced process-level separation can apply policies 2.1 times faster at system boot-up, and dynamic access control can provide secure IDCs with an overhead of only 4% for IDC system calls. We implemented SELinuxes on an ARM-based multiprocessor

The FIDES platform with multi-grained separation mechanisms is more secure than other platforms, such as software-only platforms and new-mode platforms, since it contains multi-grained separation mechanisms. Also, the above evaluation results show that four new technologies to support the FIDES platform are well suited to secure next generation mobile terminals. In future work, the FIDES platform will support SMP systems.

5. ACKNOWLEDGMENTS

Authors thank Hiroshi Chishima, Yoshiyuki Ito and all the others involved in our project for their help and technical advice. Also, we would like to express our deep appreciation for great contributions from Satoshi Suzuki, Toshiya Matsui, Masakazu Yamashina, Masajiro Fukunaga, Kouji Maeda, Naoki Nishi, Naotaka Sumihiro, and Masao Fukuma.

6. REFERENCES

[1] Alves, T. and Felton, D. *TrustZone: Integrated Hardware and Software Security.* White Paper, July 2004.

[2] ARM. *AMBA Specification (Rev 2.0).* May 1999.

[3] Bird, T. Methods to Improve Bootup Time in Linux. In *Proceedings of the Ottawa Linux Symposium*, 2004, 79-88.

[4] BREW. *The Road to Profit is Paved with Data Revenue.* QUALCOMM Internet Services White Paper, June 2002.

[5] CELF. *CE Linux Forum 1.0 Specification.* June 2004.

[6] Chaoui, J. et al. *OMAP™: Enabling Multimedia Applications in Third Generation (3G) Wireless Terminals.* Technical White Paper, December 2000.

[7] Dike, J. A user-mode port of the Linux kernel. In *Proceedings of the 4th Annual Linux Showcase & Conference,* 2000, 63-72.

[8] Gong, L. and Ellison, G. *Inside Java 2 Platform Security: Architecture, API Design and Implementation (The Java Series)*. Addison-Wesley, 2003.

[9] ITRON committee. *ITRON 4.0 Specification.* June 1999.

[10] Loscocco, P. and Smalley, S. Integrating Flexible Support for Security Policies into the Linux Operating System. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference (FREENIX '01),* 2001, 29-42.

[11] NTT DoCoMo, IBM, and Intel Corporation. *Trusted Mobile Platform Hardware Architecture Description.* June 2004.

[12] Sakai, J. et al. Multi-Tasking Parallel Method on MP211 Multi-core Application Processor. In *Proceedings of the IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips VIII),* April 2005, 198-211.

[13] Sanders, P. *Creating Symbian OS phones.* White Paper, April 2002.

[14] Semiconductor Industry Association et al. *International Technology Roadmap for Semiconductors 2003 Edition, Executive Summary.* 2003.

[15] Sugerman, J., Venkitachalam, G., and Lim, B. Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor. In *Proceedings of the 2001 USENIX Annual Technical Conference*, 2001, 1-14.

[16] Torii, S. et al. A 600MIPS 120mW 70uA Leakage Triple-CPU Mobile Application Processor Chip. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers*, February 2005, 136-137.